Evaluasi Keamanan Penyimpanan Password Menggunakan Algoritma Hash: MD5, SHA-1, dan Bcrypt

Raden Dafi Al Azhar[1*], Ina Sholihah Widiati[2]

[1]S1 Informatika
STMIK Amikom Surakarta
[1*]Radendafi123@gmail.com

[2]S1 Informatika
STMIK Amikom Surakarta
[2]inasw@dosen.amikomsolo.ac.id

Abstrak— **Keamanan password adalah konsep dasar utama dalam sistem informasi modern salah satunya yaitu penggunaan algoritma Hash, algoritma ini memungkinkan pengonversian kata sandi menjadi satu arah yang tidak mungkin lagi bisa kembali lagi ke bentuk asal. Penelitian ini dimaksudkan untuk mengevaluasi tiga algoritma hash yang secara umum digunakan yaitu: MD5, SHA-1, Bcrypt. Penulis akan menggunakan metode studi literatur dan experiment sederhana menggunakan Python untuk mengukur waktu kecepatan hashing berdasar keamananya terhadap serangan *brute force* dan *collision*. Penelitian ini menghasilkan walau MD5 dan SHA-1 sangat cepat namun tingkat keamanannya rendah, sedangkan Bcrypt waktu hashingnya lebih lambat namun dapat dibuktikan keamanannya jauh lebih kuat sehingga aman jika digunakan untuk menyimpan password.**
Kata kunci— Keamanan, Hashing, MD5, SHA-1, Bcrypt

Abstract— **Password security is a fundamental concept in modern information systems, one of which involves the use of hash algorithms. These algorithms convert passwords into a one-way form that cannot be reversed back to the original. This study aims to evaluate three commonly used hash algorithms: MD5, SHA-1, and Bcrypt. The author uses a literature review method combined with a simple experiment using Python to measure hashing speed and its resistance to brute-force and collision attacks. The results show that while MD5 and SHA-1 offer fast hashing performance, their security levels are low. In contrast, Bcrypt produces slower hashing times but provides significantly stronger protection, making it a safer choice for password storage.**
Keywords— Security, Hashing, MD5, SHA-1, Bcrypt

I. PENDAHULUAN

Pada zaman ini keamanan informasi digital semakin menjadi perhatian karena semakin maraknya kasus pencurian data, hacking atau perentasan sistem, pelanggaran privasi akun, dan lain-lain. Biasanya titik rawan dalam sistem informasi ada pada cara penyimpanan password pada server, contoh jika password disimpan pada text biasa atau *plain text* sangatlah mudah bagi pencuri mengambilnya. Keamanan password akan menjadi masalah besar ketika menyebabkan kebocoran data sensitif. Berdasarkan laporan Kominfo tahun 2023, lebih dari 60% kejahatan siber di Indonesia berasal dari kelemahan autentifikasi pengguna oleh karena itu proses penyimpanan password yang aman sangatlah dibutuhkan. Menurut Giffary dan Ramadhani [1], banyak aplikasi di Indonesia belum menerapkan sistem hashing yang aman, sehingga kerentanan sistem terhadap peretasan semakin tinggi.

Salah satu solusi yang paling sering digunakan yaitu sistem Hashing yaitu proses mengubah kode password menjadi menjadi kode tetap menggunakan fungsi satu arah yang tidak bisa dibalik kembali. Cara ini sangat efisien karena dapat membandingkan input password dengan hash yang disimpan tanpa harus menyimpan bentuk password yang asli. Terdapat tiga algoritma hash yang sering digunakan yaitu MD5, SHA-1, dan Bcrypt namun ketiganya memiliki faktor keamanan yang berbeda dan menunjukkan kelemahan terutama pada kasus collision dan serangan brute-force terutama pada MD5 dan SHA-1 ([5], [8]). Yafie [2] menambahkan bahwa sistem autentikasi modern membutuhkan algoritma yang memiliki daya tahan tinggi terhadap serangan komputasi paralel dan GPU acceleration.

Collision adalah jenis serangan kriptografi di mana penyerang mencari dua input berbeda yang menghasilkan hash yang sama dari suatu

algoritma hash sedangkan brute-force adalah metode serangan di mana penyerang mencoba segala kemungkinan pada kombinasi password menggunakan cara yang sistematis sampai menemukan yang benar, dengan kata lain coba terus hingga berhasil.

Dengan itu Bcrypt dibuat sebagai solusi dengan menerapkan salting dan cost factor yang dapat memperlambat proses hashing untuk memperkuat proteksi terhadap serangan ([1], [4]). Firdaus [6] menekankan bahwa penggunaan salt secara acak dapat mencegah penyerang menggunakan rainbow table untuk menyerang sistem. Isnainia et al. [4] menjelaskan bahwa salting mencegah dua password identik menghasilkan hash yang sama, sehingga meningkatkan keamanan secara signifikan. Selain itu, Liauren et al. [3] menyatakan bahwa peningkatan cost factor akan menyebabkan jumlah iterasi meningkat secara eksponensial, yang membuat proses hashing semakin lambat dan aman dari brute-force. Sedangkan Nasution [8] dan Zayana et al. [9] menekankan pentingnya menghindari algoritma seperti MD5 dan SHA-1 karena sangat rentan terhadap collision, yaitu kondisi di mana dua input berbeda menghasilkan output hash yang sama.

Penelitian ini akan mengevaluasi ketiga algoritma hash dengan cara membandingkan kecepatan dan keamanan sebagai faktor kelayakan jika digunakan pada sistem modern dengan cara mengkaji studi literatur dan melakukan uji coba langsung menggunakan Python.

II. METODOLOGI PENELITIAN

A. *Studi Literatur*

Penelitian ini menggunakan 10 jurnal penelitian sebagai referensi untuk memahami cara kerja Hashing, kelebihan kekurangan, dan sumber acuan penelitian ini. Penulis menemukan beberapa kelemahan MD5 dan SHA-1 terhadap Collision ([8], [9]), Beberapa literasi ini meliputi aspek implementasi Bcrypt pada sistem web ([1], [3], [4], [10]), efektivitas cost factor ([2], [4]), serta analisis kerentanan MD5 dan SHA-1 terhadap serangan ([5], [7], [8], [9]).

B. *Eksperimen Uji Coba Hashing*

Penulis melaksanakan eksperimen menggunakan metode uji coba sederhana menggunakan bahasa pemrograman Python untuk mengukur waktu rata-rata kecepatan Hashing menggunakan algoritma MD5, SHA-1, dan Bcrypt (bcrypt package). dengan password ”ujipassword01” sebanyak 10 kali, yang nantinya akan dihitung rata-rata deviasi waktunya masing-masing.

III. HASIL DAN PEMBAHASAN

A. *Studi Literatur*

Berdasarkan literatur, algoritma MD5 dan SHA-1 memiliki kelemahan signifikan terhadap brute-force dan collision karena hasil hash-nya bersifat deterministik dan sangat cepat. Nasution [8] menjelaskan bahwa algoritma SHA-1 telah terbukti rentan terhadap serangan kolisi sejak ditemukannya teknik SHAttered oleh Google. Zayana et al. [9] mendukung hal ini dengan menunjukkan bahwa hash MD5 dapat dengan mudah dibalik menggunakan teknik rainbow table, terutama ketika digunakan tanpa salt. Isnainia et al. [4] menyebutkan bahwa collision menjadi sangat berbahaya ketika penyerang mampu memalsukan data otentik dengan hash identik.

Sebaliknya, algoritma Bcrypt dikembangkan dengan mempertimbangkan aspek keamanan autentikasi secara khusus. Giffary dan Ramadhani [1] menunjukkan bahwa Bcrypt menggunakan proses salting sepanjang 128-bit yang mampu menghasilkan hash unik untuk input yang sama. Liauren et al. [3] juga menambahkan bahwa fitur cost factor dalam Bcrypt dapat disesuaikan agar tetap relevan menghadapi peningkatan daya komputasi perangkat keras. Semakin tinggi cost factor, semakin besar pula jumlah iterasi hashing yang harus dilakukan, sehingga memperlambat proses peretasan secara signifikan ([2], [3]).

Putra et al. [7] menyatakan bahwa Bcrypt lebih tahan terhadap berbagai metode serangan seperti brute-force berbasis tools otomatis seperti Hydra, Medusa, dan Ncrack. Bahkan ketika diuji

terhadap kata sandi yang sama, hasil hash Bcrypt tetap berbeda berkat salting dinamis, seperti dijelaskan oleh Isnainia et al. [4]. Selain itu, Firdaus [6] menyoroti bahwa kombinasi antara salt dan cost factor dalam Bcrypt memperlambat waktu respons per tebakan, sehingga sangat menghambat brute force dalam jangka panjang.

Yafie [2] menyimpulkan bahwa Bcrypt merupakan pilihan ideal untuk sistem autentikasi berbasis web karena dapat diatur sesuai kemampuan server tanpa mengorbankan keamanan. Dukungan framework modern seperti Laravel terhadap algoritma ini juga memperkuat kemampuan pengaplikasian (aplikabilitas) dalam pengembangan sistem [10].

B. Eksperimen Uji Coba Hashing

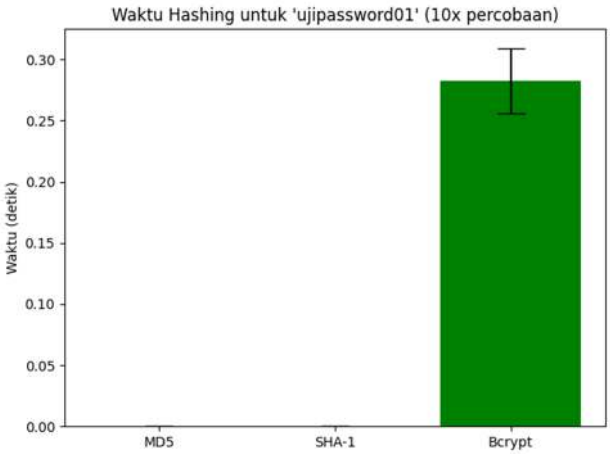
```python
# Password untuk diuji
password = b"ujipassword01"
iterations = 10
results = {"MD5": [], "SHA-1": [], "Bcrypt": []}

# Eksperimen hashing
for _ in range(iterations):
    start = time.perf_counter()
    hashlib.md5(password).hexdigest()
    results["MD5"].append(time.perf_counter() - start)

    start = time.perf_counter()
    hashlib.sha1(password).hexdigest()
    results["SHA-1"].append(time.perf_counter() - start)

    start = time.perf_counter()
    bcrypt.hashpw(password, bcrypt.gensalt())
    results["Bcrypt"].append(time.perf_counter() - start)

# Hitung rata-rata dan deviasi
avg = {algo: statistics.mean(results[algo]) for algo in results}
std = {algo: statistics.stdev(results[algo]) for algo in results}
```

Gambar 1. Kode eksekusi eksperimen Hashing

Dengan menggunakan kode pada Gambar 1 dapat menghasilkan Tabel 1:

Tabel 1. Rata-rata dan Deviasi Standar Waktu Hashing Password "ujipassword01"

Algoritma	Rata-rata Waktu (detik)	Deviasi Standar
MD5	0,00002	0,00000
SHA-1	0,00001	0,00001
Bcrypt	0,28270	0,02665



Gambar 2. Grafik statistik waktu hashing setelah 10× uji coba

Pada gambar 2 dapat dilihat, grafik pada gambar menunjukkan:

1. MD5 Memiliki rata-rata waktu hashing tercepat: 0.00002 detik dengan deviasi standar sangat kecil, menunjukkan hasil sangat konsisten.
2. SHA-1 Sedikit lebih lambat dari MD5, tetapi masih sangat cepat: 0.00001 detik deviasi sedikit lebih besar, tetapi masih tergolong stabil dan cepat
3. Bcrypt dengan rata-rata waktu hashing tertinggi: 0.28270 detik deviasi standar relatif besar (±0.02665 detik) karena proses gensalt() atau *Generate salt* menghasilkan variasi waktu berbeda di setiap eksekusi.

Grafik ini memperjelas perbedaan yang mencolok antara algoritma tradisional seperti MD5 dan SHA-1 yang sangat cepat namun tidak aman, dibandingkan dengan Bcrypt yang jauh lebih lambat tetapi secara signifikan lebih aman.

Waktu eksekusi Bcrypt yang tinggi bukanlah sebuah kelemahan, tetapi menjadi fitur keamanan, karena memperlambat proses penyerangan brute-force secara drastis.

Setelah di uji coba dan dianalisis Seperti yang ditunjukkan pada Tabel 1, MD5 dan SHA-1 memiliki waktu hashing yang jauh lebih cepat dibandingkan Bcrypt. Hal ini justru menjadi kelemahan karena memungkinkan jutaan percobaan brute-force dalam waktu singkat ([5], [7]). Sebaliknya, Bcrypt membutuhkan waktu lebih lama, namun justru hal ini menjadi

keunggulan karena menyulitkan serangan berulang ([2], [6]). Setiap delay pada Bcrypt membuat brute-force secara masif menjadi tidak efisien.

Isnainia et al. [4] menyatakan bahwa dengan meningkatnya *cost factor*, jumlah iterasi akan bertambah secara eksponensial sehingga menambah beban komputasi pada setiap proses hashing.

IV. KESIMPULAN

Dari hasil penelitian melalui uji coba hash dan studi literatur dapat disimpulkan jika: MD5 dan SHA-1 sudah tidak lagi layak dipakai kedalam sistem penyimpanan password karena keduanya terbukti terlalu cepat dalam proses hashing-nya, yang justru melemahkan tingkat keamanannya. Berkebalikan dengan Bcrypt hasil experiment sudah membuktikan dengan kecepatannya yang lambat Bcrypt dapat memberikan keamanan dan perlindungan yang lebih maksimum. Dengan ini penulis dapat merekomendasikan penggunaan Bcrypt sebagai standar minimum penyimpanan password.

UCAPAN TERIMA KASIH

Penulis mengucapkan terimakasih kepada dosen pembimbing, peneliti yang referensinya saya gunakan, orang tua, penyelenggara Senatib dan semua pihak yang membantu dan mendukung proses berlangsungnya penelitian hingga pembuatan jurnal ini.

REFERENSI

[1] R. S. Giffary and E. Ramadhani, "Implementasi Bcrypt dengan SHA-256 pada Password Pengguna Aplikasi Golek Kost," *J. Sist. Komput. dan Inform.*, vol. 3, no. 4, p. 543, 2022, doi: 10.30865/json.v3i4.4285.

[2] H. Naufal Yafie, "Analisis Penggunaan Fungsi Hash BCrypt untuk Keamanan Kata Sandi," 2020.

[3] R. M. Liauren, B. Zaman, and S. Bahri, "Implementasi Algortima Aes Dan Bcrypt Untuk Pengamanan Data Pengguna Pada Website Jahitku," *KHARISMA Tech*, vol. 20, no. 1, pp. 57–71, 2025, doi: 10.55645/kharismatech.v20i1.535.

[4] K. Nur, D. Suhartono, M. Thoriq, and A. Qothrunnada, "Implementasi Pengamanan Data Menggunakan Teknik Bcrypt Hashing Password dan Algoritma Advanced Encryption Standard (AES)," *J. Sist. dan Teknol. Inf.*, vol. 13, no. 1, pp. 101–108, 2025, doi: 10.26418/justin.v13i1.84997.

[5] A. Andilala, A. K. Hidayah, A. W. Mahfuzy, and M. Oki, "Implementasi Kombinasi Enkripsi Base64 Dengan Hashing Sha-1 Dan Md5 Pada Aplikasi Perpustakaan Universitas Muhammadiyah Bengkulu," *J-SISKO TECH (Jurnal Teknol. Sist. Inf. dan Sist. Komput. TGD)*, vol. 6, no. 2, p. 694, 2023, doi: 10.53513/jsk.v6i2.8546.

[6] F. M. Risqi, "Analisis Penggunaan Algoritma Bcrypt dengan Garam (Salt) untuk Pengamanan Password dari Peretasan," *Makal. IF2120 Mat. Disk.*, p. 1, 2022, [Online]. Available: Analisis Penggunaan Algoritma Bcrypt dengan Garam (Salt) untuk Pengamanan Password dari Peretasan

[7] Dewa Made Julijati Putra, I Nyoman Namo Yoga Anantra, Putu Adhitya kusuma, Putu Damar Jagat Pratama, Gede Arna Jude Saskara, and I Made Edy Listartha, "Analisis Perbandingan Serangan Hydra, Medusa Dan Ncrack Pada Password Attack," *J. Inform. Teknol. dan Sains*, vol. 4, no. 4, pp. 461–466, 2022, doi: 10.51401/jinteks.v4i4.2192.

[8] R. M. Nasution, "Implementasi Metode Secure Hash Algorithm (SHA-1) Untuk Mendeteksi Orisinalitas File Audio," *Bull. Comput. Sci. Res.*, vol. 2, no. 3, pp. 73–84, 2022, doi: 10.47065/bulletincsr.v2i3.140.

[9] M. Rafiiq Zayana, I. Fitri, and A. Gunaryati, ") 2022 2,3,4 Program Studi Sistem Informasi, Fakultas Teknologi Komunikasi dan Informatika," *J. Teknol. Inf. dan Komunikasi)*, vol. 6, no. 3, 2022, [Online]. Available: https://doi.org/10.35870/jti

[10] D. Febrian *et al.*, "Implementation of Bcrypt Algorithm on Website-Based Hashing Generator Using Laravel Framework," *J. Inf. Syst. Informatics Comput.*, vol. 7, no. 2, p. 199, 2023, doi: 10.52362/jisicom.v7i2.1130.